**Introduction**

The main part of this project is the fight scene between the player and the monster ai. That will be the part where the user will find entertainment and find the purpose of the game. Intro story scene builds up the curiosity from the user, and the actual playing scene should fulfil the player’s entertaining needs. To accomplish such goal, the fight scene should be leveled challenging yet easy enough for the player to clear it after some amount of trials and practice. With that being said, the balance between player character’s power and the monster ai’s power is crucial so the interaction between them is tightly going back and forth. In more details, the monster ai’s attack shouldn’t be unavoidable but difficult enough for the player to focus and dodge. The important factors that will affect this system are vector velocity of the attack prefab launched by the monster traveling in the scene space. Also, the float value of its damage should be modified to fit the balance. The same thing goes for the player game object, that the player’s character has to be controlled to move fast enough to dodge the monster’s attack yet not too fast breaking the balance, and attack prefab launched by the player should pass proper float value of damage to monster ai script to process its hp reduction.

Audio is also one of the key factors to bring the game scene alive. There are too many audio sources to be used in the game scene for attacks, movement, UI, and background music, and they need to be played at the proper time and ends correctly depending on the situation. To accomplish the such goals, instead of attaching each audio file to each game prefab/object/scene, I create the audio manager script to manage all the audio sources at once.

Since the game scene is all about the interactions among the objects/prefabs, it is important to write the behavior/method script for each component carefully, considering the later referencing and passing the value/information to bring the dramatic changes in the scene. Carefully classify private and public variable/method to accomplish it.

Safety and security are never a factor that will affect this offline-based single user game, since the game will not require the player to type in any personal information or confidential data to login or anything. Also, performance of the software is merely a factor to be worried since the game does not use any big memory space or network to be played.

This is a single man project, thus communication with other project member or modification caused by difference of opinions are neglectable.

Maintainability of general games is managed by online website or game distributor platform, however this project is an offline-game that will be distributed through personal website so it is impossible to respond to bug reports or further user requirements from the users. There are two ways to resolve this.

1. If there is any more level design demands or bug reports from the user simply modify the game project on Unity and rebuild and re-distribute the game file to the users.
2. No further update or maintenance will be made but announce that this game is a prequel of bigger game project coming up and use it as advertisement purpose.

**Architecture and Design Philosophy**

There were many possible modes of game that it would be designed into in the proposal stage of the development.

1. Multiple level designs leading up to the boss stage and repeats the process until the last stage boss to see the ending of the game.
2. Player vs. player style game that a player can participate in the host player’s server, and these two players play against each other.
3. Role-playing based game that fights the monster in multiple level designs which a player can acquire game items and evolve the strength to fight the stronger monster in the later stage.

However, those game project would require a single developer myself way longer time than the given time. Therefore, I had to discard most of the considered game parts and leave the major key factor to keep the project simple environment. I decided to leave only one level of boss stage and make it an offline single player game. The game should not be too simple and boring, thus I added other features that can bring diverse experience to the game such as healing zone for the player, special phases for both of the monster ai and the player, and a timer to make the player feel challenging to clear the stage.

If there is a demand of deeper level design in the game or more various game features like items and skills, they will be developed and added to the project on Unity and the game will be rebuilt. It is also possible to reference the IP of this game project to create whole another game in different style. Even network based game style can be combined into this project later in the future, and that is the advantage of creating the game in Unity engine that it gives flexibility in modifying and updating the game

**Architectural Views**

**[Logical View]**

The game shall provide interesting story and proper amount playing time to the user to be entertained. The game receives key inputs from the player and should display interesting interaction between the player object and other objects/scenes in the game scene. It is important to deliver unique and convenient controls over the player object thus the player does not easily get tired of the content.

Key inputs

Game

Player

Graphical Response

**[Process View]**

The player object, the monster ai, and game scenes will continuously interact with each other throughout the playing time. Main interactions between the player object and the monster ai are tracking their local position on the game scene space and trading float variable named damage to track the circumstance of the current fight. Each game scene (main menu, game over, you win, boss stage, and splash screen) should interact with each other in a way that one scene loads another scene under particular circumstances. This flow of scene transition is crucial to convey smooth and immersing atmosphere throughout the whole game. Also, there’s an interaction between the user and the game scenes/objects as they receive the user input to bring next action to the game.

Passes the local position

Player object

Monster ai

Lerp to the player’s local position

Player

dies

Monster dies

Instantiate

Instantiate

Pass the damage value

Pass the damage value

Attack prefab

Game over

You win

Main menu

User Input

**[Development View]**

Each script, to pass the damage to either player or monster ai, inside the game prefab shares similar format with just a bit of modification. Similarly scene layouts and scripts to control them are similar to each other. For example, there are two different attack prefab belongs to the player which are “regular bullet prefab” and “sniper bullet prefab.” They use exactly the same code to be instantiated at the player object’s fire point and travel along x-axis passing damage value to the monster ai if it collides to the monster object. Similarly attack prefabs belong to monster ai have similar code to deliver damage to the player object when its in contact with it, however, they have more various and dramatic pattern to carry out the job, thus they only follow one main format and get modified based on their purpose. As well as the scenes have same layout where they have buttons and graphical interfaces for the player’s convenience, but each component has different function and positioned differently.

And more…

Monster ai attack prefabs

Bullet prefab script

Regular bullet prefab

‘Bloodshower’ prefab

‘Hellfire’ prefab

‘Eye bullet’ prefab

Vector velocity = 15

Damage value range

= 5 - 10

-

Sniper bullet prefab

Damage = 15

\*Free fall from the instantiate point

\*Multiple instantiation

Travels with ai

Damage = 5

\*constantly passing damage to the player in intervals

Vector velocity = 20

Damage range

= 15 – 25

Vector velocity = 20

Damage range

= 15 - 25

**[Physical View]**

The game is developed on game engine (Unity) thus it is simple work to reference image files, animation, audio source, game objects, and user input inside the code. There are built-in libraries to help the developer to utilize external components in programming language (C#). You can always modify few settings like graphic rendering, scale, audio volume, user key input references, and animation on the editing window.

**Design Models**

<<System>>

Game over

**[1. Context Model]**

<<System>>

Pause menu

<<System>>

Easter egg

<<System>>

Splash Screen

<<System>>

You win

<<System>>

Boss Stage

<<System>>

Main menu

<<System>>

Game

**[2. Interaction Model]**

You win

Game over

Splash Screen

Main menu

Boss Stage

Player

Scene Change

Play button

Run the app

Player wins

Go back to menu

Play again

Go to main menu

Player winss

Play again

**[3. Structural Model]**

Game Objects

Scenes

Monster AI

Main menu

Splash Screen

Boss Fight

Player Object

User Interface

Pause Menu

Grid map

Light Source

Play button

Menu button

Exit button

C

Audio Manager

Scene Manager

Story Text

Credits Text

Movement Script

Attack Script

2D rigid body

Collider

2D rigid body

Collider

Movement Script

Attack Script

Animation

Game over

You win

Game prefabs

Sprites

Audio sources

Player Attacks

Monster Attacks

Etc..

Play button

Exit button

Play button

Exit button

Score display

**[4. Behavioral Model]**

Run

Exit the game

Main menu

Intro Story

Splash Screen

Go to menu

Displays Scores

Reset Scores

Exit

Menu

Exit

Menu

Play

Game Over Scene

Play

Time over

Player dies

Easter Egg Scene

You win scene

Killing the monster ai

Clear time > 3min

Clear time < 3min

Score

Exit

Play